



GRASP

(Grid Resource Allocation Services Package)

Administrator's Guide

KMI-R1 Developer Team <kmi@moredream.org>

Document Version: 1.0 Date: November 07, 2005



Copyright 2002-2005 Korea Institute of Scientist and Technology Information.
All rights reserved.

This document is licensed under the terms of the K*Grid Public License.
The details of K*Grid Public License is found at
<http://kmi.moredream.org/downloads/license.html>.

Contents

1	Introduction.....	5
1.1	What is GRASP?	5
1.2	Architecture and Components of GRASP	5
1.2.1	Overview	5
1.2.2	JSS (Job Submission Service).....	8
1.2.3	GSS (Grid Scheduling Service)	9
1.2.4	RMS (Resource Manager Service)	11
1.2.5	RRS (Resource Reservation Service)	13
1.2.6	SRB enabled globus-url-copy	14
1.2.7	Client Tools.....	16
1.3	Getting help	17
2	Installation and Configuration	18
2.1	Support software	18
2.1.1	Required.....	18
2.1.2	Optional.....	18
2.2	Installing support softwares	18
2.2.1	Installing Java SDK.....	18
2.2.2	Installing Globus Toolkit.....	18
2.2.3	MySQL Database.....	19
2.2.4	MySQL-Connector/J 3.0 (JDBC Driver).....	19
2.2.5	PHP4	20
2.2.6	PHP4 + Apache Web Server (optional)	20
2.2.7	OpenPBS and Cluster Configuration.....	21
2.3	Installing GRASP	22
2.3.1	JSS (Job Submission Service).....	22
2.3.2	GSS (Grid Scheduling Service)	22
2.3.3	RMS (Resource Manager Service) and RRS (Resource Reservation Service) 23	
2.3.4	Client Tools.....	26
2.4	Testing.....	27
2.4.1	Running the first job	27
3	Operation of GRASP services	29
3.1	JSS	29
3.1.1	Logging	29

3.2	GSS	29
3.3	RMS	30
3.3.1	Logging	30
3.4	RRS	30
3.4.1	Logging	30
3.4.2	Database Connection	30
3.4.3	Monitoring Reservation Status	31
4	References	33

1 Introduction

This document is intended to provide the system administrators with the information required to install, configure, and manage GRASP.

1.1 What is GRASP?

The problem of Grid resource allocation is concerning about delivering the users distributed resources with computing powers, data storage capacity, network connectivity, etc. The Managed job service in Globus toolkit 3.x (GT3) is the service to be used to run the job on a remote resource. However, in order to build more useful Grid, there should be added some user-friendly features and advanced resource allocation techniques including resource brokering, scheduling, job monitoring, and so forth. To meet this requirement in Grid resource management area, we designed and implemented a resource allocation system named GRASP(Grid Resource Allocation Services Package), which is to let users to submit their jobs in more efficient and intelligent manner to the Grid resources. The services of GRASP were implemented based on the OGSi specification implementation of GT3 as well as other services in MoreDream. Followings are brief introduction of GRASP functionalities.

1.2 Architecture and Components of GRASP

1.2.1 Overview

GRASP supports scientific applications with the high performance computing features such as MPI, high throughput computing features such as parametric studies, and data intensive features. GRASP can handle three kinds of job type: SINGLE, XMPI, and HTC. SINGLE is a simple job to use only one computing node. XMPI is an MPI job which can be run over multiple resources. Lastly, HTC is a job for HTC applications such as parametric study. In Both XMPI and HTC job case, GRASP co-allocates multiple resources to the job even though the resources are remotely distributed. To support data intensive features, we added the feature to automatically stage in files from SRB server and stage out the files to SRB server

Furthermore, we have designed the job description language, named JRDL (Job and Resource Description Language) to overcome the limitation of the GT3. RSL2, the GT3 job description language, just describes job specifications rather than resource specifications such as resource preference. RSL2 is also not considering about co-allocation. Therefore, we have proposed JRDL to meet both requirements for the job and resource's preference. The resource preference part is used in matchmaking step in Grid scheduling service (GSS). JRDL is

designed based on XML schema.

GRASP is composed of four useful services needed to allocate the resources in Grid as illustrated in Figure 1. Firstly, the resource brokering is done by Grid scheduling service (GSS). This service finds out resources from the index service which are fit to a user's job and then reserve the resources in advance through Resource reservation service (RSS). To select proper resources it performs matchmaking between the resource specification from the user and the job/user specification preferred by the resource administrator. And then the resources are allocated to the job. Secondly, Job submission service (JSS) does co-allocation of resources and co-monitoring of the job. Co-allocation in GRASP makes it possible the job submission to the multiple distributed resources simultaneously. And co-monitoring allows the user to monitor her job flow. Lastly, Resource manager service (RMS) authenticates the user for the job execution on a local resource and submits the job to the local batch queuing system such as PBS. RMS should get the permission to allocate resource from RRS before submitting the job. Followings are the main features of GRASP, job types that is handled by GRASP and job statuses defined in GRASP. The explanation of each service, JSS (Job Submission Service), GSS (Grid Scheduling Service), RMS (Resource Manager Service), and RRS (Resource Reservation Service), will be followed after this overview section.

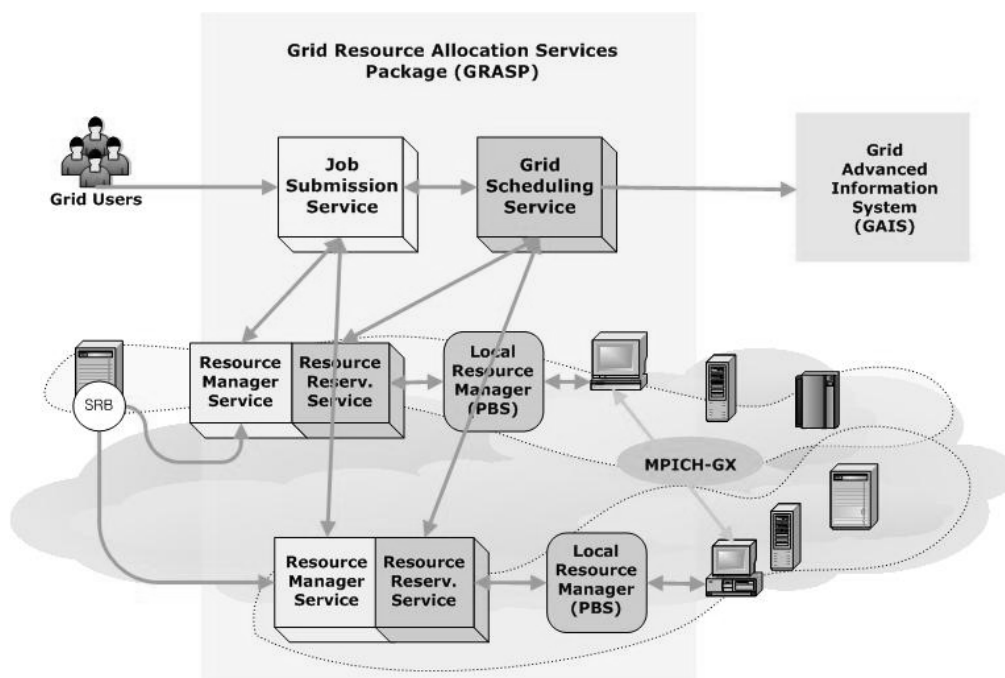


Figure 1. Architecture of GRASP

Main Features

- All services are OGSI-compliant Grid services.
- GRASP supports three kinds of job type: SINGLE, XMPI, and HTC. SINGLE is a simple job which uses only one computing node. XMPI is an MPI job which can be run over multiple resources. Lastly, HTC is a job for high throughput computing such as parametric study.
- Multiple resources can be co-allocated to a job even though the resources are remotely distributed.
- Scheduler can automatically select resources by matchmaking process.
- Job can reserve resources in advance.
- The input files can be staged in from SRB server and the output files can be staged out to SRB server automatically.
- We provide JRDL (Job and Resource Description Language) as a general language to describe a job and user preferences required allocating resources for a job in Grid environment.
- We bring client tools for job creating, submission, controlling, and monitoring. They provide three user interfaces having same functionality: a command line interface, a graphic user interface, and web interface.

Job Type

We are supporting three kinds of job type: SINGLE, XMPI, and HTC.

- SINGLE: It is a simple job which uses only one computational node (e.g. simple script for pre/post processing).
- XMPI: It is an MPI job which uses multiple resources to run even though resources are remotely distributed. Each resource could have several nodes.
- HTC: It is a job which uses multiple resources to run and have no communication between each of all subjobs (e.g. parametric study). Each subjob must be a SINGLE job.

Job Status

Job Submission Status

Job submission service manages the status of job submission. The Status has following information:

- State of job
- All subjobs' statuses
- Fault message.

Job State

- "Unsubmitted": JRDL is unsubmitted to Job submission service.
- "Scheduling": Job is scheduling to find proper resources at Grid scheduling service.
- "Pending": Job is pending even though the subjob is submitted to Resource manager service.
- "Active": Job is active.
- "Suspended": Job is suspended.
- "Done": Job is done.
- "Failed": Job is failed.

Subjob Status

Resource manager service manages the statuses of subjobs. Each status has following information:

- Subjob id
- State of subjob state
- Execution time of subjob: start time and end time of job
- Allocation information of subjob: allocated resources' address
- Fault message.

Subjob State

- "Unsubmitted": Subjob is unsubmitted to ResourceManagerService.
- "StageIn": Subjob is staging in the files to need to execute.
- "Waiting": Subjob is waiting for its requested execution time to be reached
- "Pending": Subjob is pending even though the subjob is submitted to the local job manager
- "XMPI_init": XMPI subjob is initializing
- "Active": Subjob is active.
- "StageOut": Subjob is staging out the files to result from executing
- "Suspended": Subjob is suspended.
- "Done": Subjob is done.
- "Failed": Subjob is failed.

1.2.2 JSS (Job Submission Service)

Key concepts

JSS is a Grid service to enable a job to submit to the resources in Grid testbed and enable a

user to monitor the status of submitted job. We provide JRDL language to describe the job, which is "an atomic task" of a workflow specification or other kinds of a complex, multi-step application. The service has the status of job submission and the requested JRDL which are provided as service data.

Architecture

The job submission process is illustrated in Figure 2. When JSS receives the JRDL, the service can determine resources by GSS which is a Grid service to find out resources which are fit to the user's job from information provider and make a reservation to RRS on each resource. User could the resources manually by specifying the address and local job manager type of resources to JRDL. If the resources are decided, the job is divided into subjobs, then that are co-allocated to RMS on each resource.

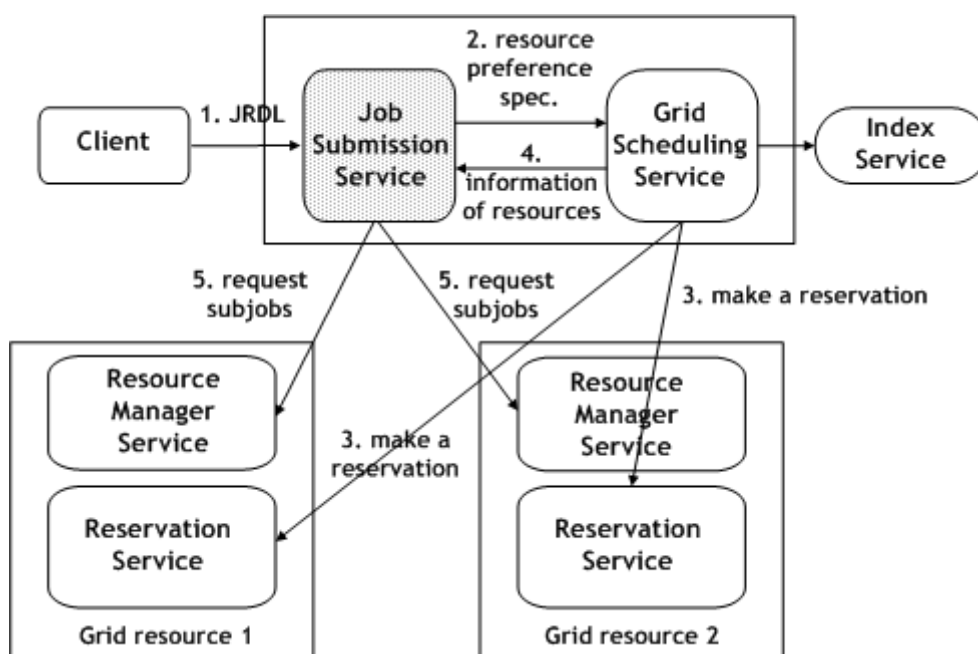


Figure 2. Job Submission Process

1.2.3 GSS (Grid Scheduling Service)

Key concepts

Grids consist of a large variety of services which reveal accessibilities to resources and the access to a resource is under control of the policies of the resource owners. Besides, complicate bottom layer Grid fabric should be hidden from Grid users. Therefore, the scheduling service which coordinates between various resources and higher level consumers satisfying policies on both sides is acutely needed in the Grid computing environment. The GSS in

GRASP was designed and implemented to do scheduling in such a complex Grid infrastructure for the jobs from various applications.

Major purpose of the GSS is to find resources which meet user's requirements and select resources according to a scheduling algorithm. In order to discover proper resources the GSS queries an information service, GAIS in MoreDream, with resource specification for the job. The GSS does screening process to choose the resources which meet minimal requirements to execute the job.

And then, with the filtered resources, selection is done by the specific scheduling policy. The Grid scheduling service can have several scheduling plugins which implement application-specific policies or scheduling algorithms. The plugin selected by the user will be applied to select most appropriate resources.

Once the selection process is done by a scheduling plugin, the service tries reservations to the selected resources for the time that user have specified in JRDL file. If the reservation fails, the service gains recent information about available resources from the reservation services, does scheduling again, and then retries reservation to the resources. These processes are repeated until the selected resources are confirmed with reservation IDs.

Architecture

The architecture of GSS is described in Figure 3. Following paragraphs explains each parts of GSS.

The GSS has the factory mechanism i.e. the GSS factory service creates a GSS instance and the created instance deals with the requested job until it gains resources. GSS acquires candidate resources thru Resource Broker. In this step, Resource Broker queries information of available resources to GAIS filtering out the unfit resources.

The scheduling plugin which was specified in the job request takes the job and resource candidates. And then it makes a map between the job and resources according to the scheduling policy. Scheduling plugin can have policies or selection algorithm. GSS has the default plugin, in which a opportunistic load balancing (OLB) algorithm is implemented. OLB assigns each task in the job, in arbitrary order, to the next available node, regardless of the task's expected execution time on that resource. In the distribution of GRASP package, HTC plugin and MPI plugin is included for each type of jobs in addition to the default plugin.

The Reservation Agent takes selected by scheduling plugin and tries to reserve resources for the certainty of the schedule. When the reservation trials are not complete, it asks available node resource capacity to the resource candidates and repeats scheduling and reservation keeping succeeded reservations.

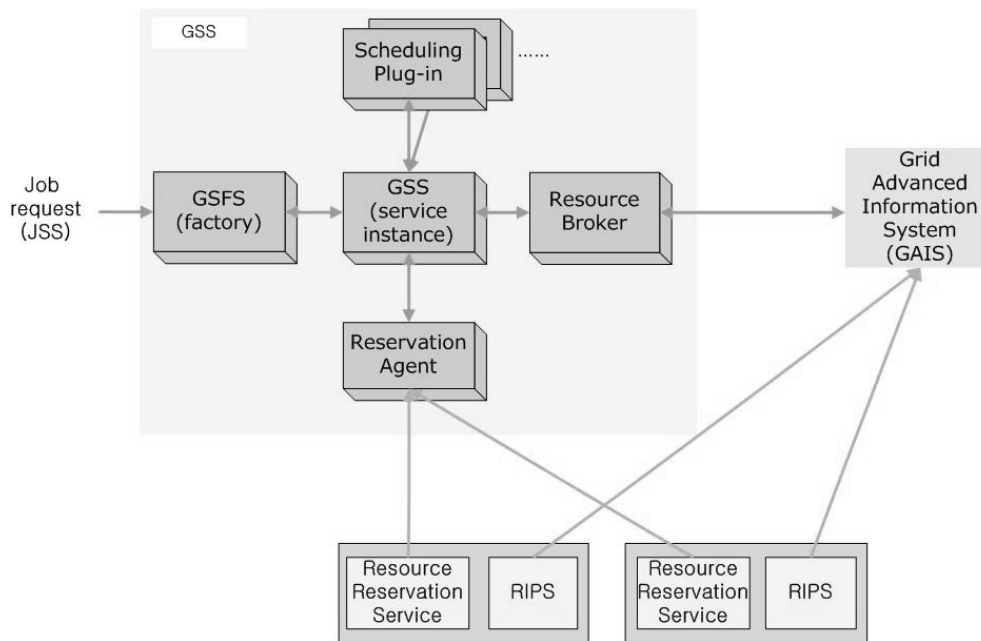


Figure 3. Architecture of GSS

1.2.4 RMS (Resource Manager Service)

Key concepts

RMS is a Grid service to enable subjobs to allocate resources and be executed by the local batch scheduler such as PBS. Resources are computational nodes to be managed by the local batch scheduler. The service has the statuses of subjobs which are provided as service data.

Architecture

Local resource allocation and execution process is illustrated in Figure 4. When RMS receives the execution request for subjobs, the service must allocate local resources. This service can allocate resources when the service gets the permission from RRS, which has established the reservation by the request of Grid scheduling service. RMS then invokes JMS (Job Manager Script) to submit the subjobs to designated local batch scheduler. While the subjobs are running, the service manages the status of each subjob. Because RMS is managed by Job submission service, this service notify the status of subjob to JSS whenever the status changes.

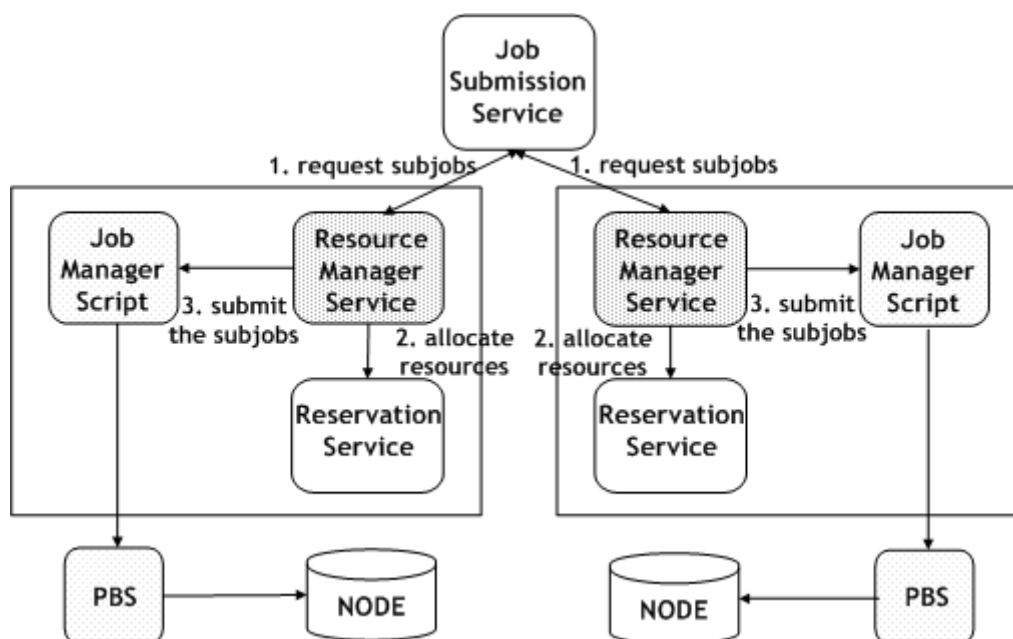


Figure 4. Local Resource Allocation and Execution Process

JMS (Job Manger Script)

JMS module processes file stage-in, file stage-out, submit, poll, and kill requests instead of RMS. Job manager script is written in PHP language, and can be executed by the java Runtime.exec() method. The requests from the RMS is delivered to JMS as an XML document form. JMS parses the XML and processes the requested function.

JMS is installed under \$GLOBUS_LOCATION/libexec/grasp-jms-php

JMS can be executed manually in a command line prompt:

```
$ jms -file < filename>
```

The filename is a path name of an XML file, which has the format:

```
<xml>
<action>ACTION</action>
<manager>MANAGER</manager>
<jobtype>JOBTYPE</jobtype>
...
</xml>
```

where,

```
ACTION = proxy_relocate | submit | poll | stage_in | stage_out
MANAGER = pbs | fork
JOBTYPE = single | htc | xmpi
```

File stage-in and stage-out actions use globus-url-copy command to copy files between local file system and remote server. To use PBS manager, OpenPBS must be installed in the local

system. The configuration file (config.php) sets up these path information.

1.2.5 RRS (Resource Reservation Service)

A Grid resource is composed of many kinds of resources, such as CPUs and memory, storage space, network bandwidth, special purpose instruments. RRS manages reservation of resources which are able to be reserved on the Grid. As a simple case, computing nodes of a cluster system is a kind of a resource which can be reserved. RRS makes a reservation to only computing nodes of a cluster.

In general, to make a reservation to resources, a user should specify the followings:

- The start time of reservation
- The end time of reservation (or duration from the start time)
- The kind of resource to reserve
- The identity of reservation maker
- Amount of resource to reserve

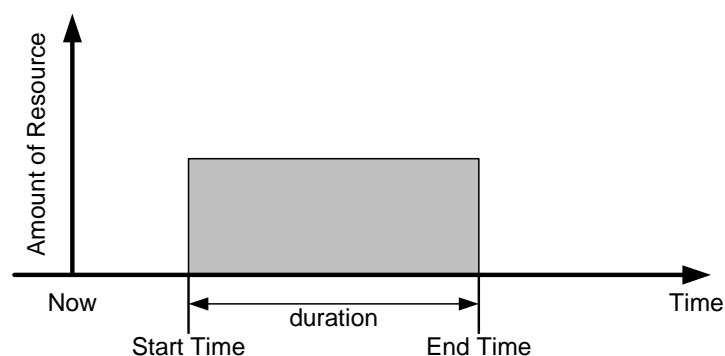


Figure 5. Reservation information: start time, duration, amount of resource, the type of resource and the identity of reservation maker

Reservation Status

Figure 6 and Figure 7 show the status change of a reservation item. It starts from the 'unknown' status. Until the start time, the reservation is in 'waiting' status, which means that the reservation is valid but it is not ready to be allocated yet. Immediately after the start time the status of the reservation changes to be 'wait_alloc' status, which means the reservation is valid and it is available for allocation now. After the job runs on the resources, the status of the reservation changes to be 'allocated'. Finally, when the job finished successfully, the status changes to be 'done'. If the valid reservation interval is over with no job allocated, the reservation status becomes to 'invalid' status. Reservations can be canceled if the status is one of 'unknown', 'waiting', or 'wait_alloc' status.

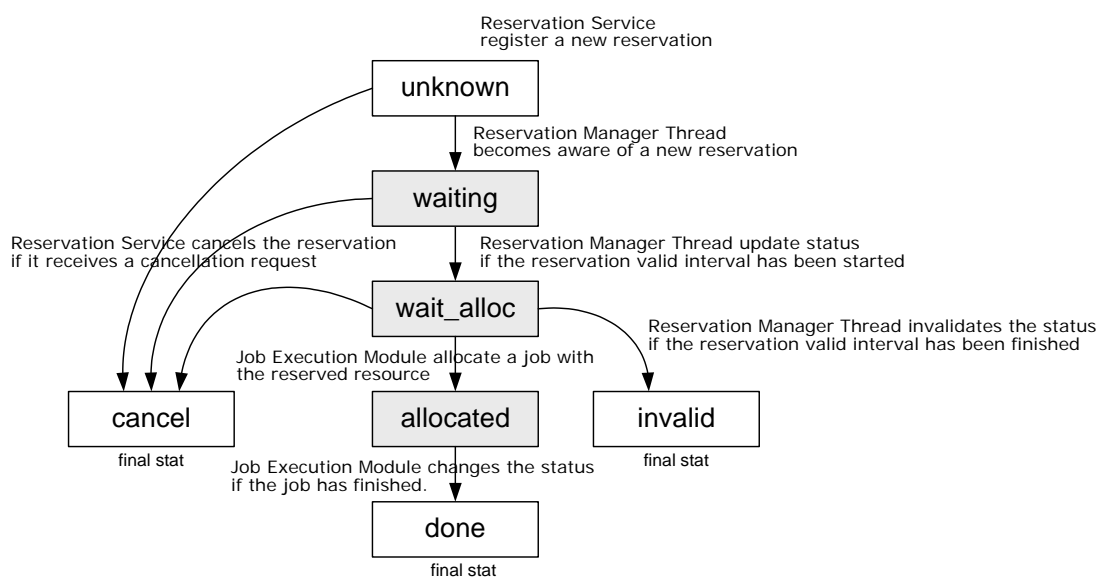


Figure 6. Status changes of a reservation

The 'cancel', 'invalid', and 'done' are final status. The 'waiting', 'wait_alloc' and 'allocated' status are valid status, which means that reserved resources are not be reserved or available by other users.

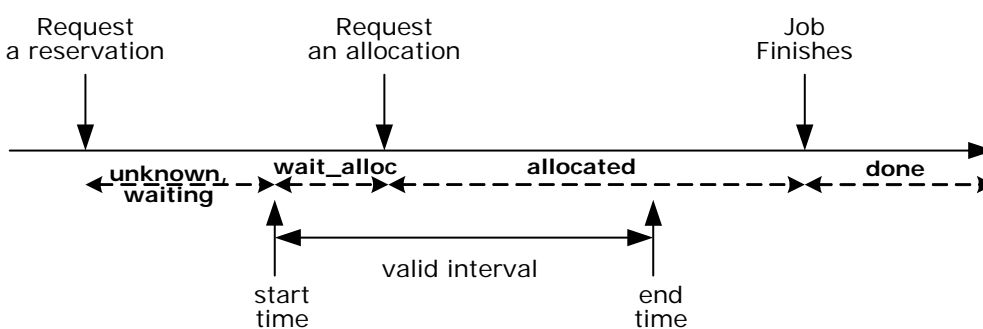


Figure 7. Reservation status changes in a time line

The Figure 7 depicts the status changes of a reservation in a time line.

1.2.6 SRB enabled globus-url-copy

Overview

globus-url-copy, which is an application of Globus toolkit, copies a file specified by source URL to a location specified by destination URL, using the GASS transfer API. It is used to stage in/out files from/to storage device for executing jobs. All protocols supported by GASS (local file, http, https, ftp, and gsiftp) are supported. Piping to/from stdin/stdout (setting source/dest argument = '-') is also supported. However, it could not retrieve/save data from/to storage not to

be able to use protocol supported by GASS. The Storage Resource Broker (SRB) is client-server middleware that provides a uniform interface for connecting to heterogeneous data resources over a network and accessing replicated data sets. SRB support a lot of interfaces for data resources including HRM, HPSS, DB2, Oracle, Illustra, ObjectStore, ADSM, UniTree, UNIX, NTFS, and HTTP. Therefore, we modify `globus-url-copy` application to support SRB protocols as well as GASS protocols for accessing various data resources and replicated data sets.

SRB URI

The location of data file should be described with URL format to use `globus-url-copy`. Thus, the location of data in SRB could be described with URI format as showed in Figure 8. Actually, both `replica` and `resource` are not included in “http://www1.ietf.org/proceedings_new/04nov/IDs/draft-gilbert-srb-uri-00.txt”. `replica` might be used for accessing replicated data sets. `resource` might be used to specify the resource name for creating new data to SRB.

```

srb:// [username.mdasdomain [.zone] [:password] @] host [:port]
[?replica=replica_id][?resource=resources_name] [/path]

```

where square brackets [...] delineate optional components, the characters `:`, `/`, `@`, and `.` stand for themselves, and spaces should be ignored. If the optional port number is not included, the default port 5544 will be used.

Figure 8. Syntax for SRB URI

As mentioned above, the data could be retrieved from SRB server or be saved to SRB server by specified SRB URI format. If there are no attributes except `replica` and `path` in specified format, default configuration will be read in `~/srb/.MdasEnv` file. The `~/srb/.MdasEnv` file includes following default configuration information to connect to SRB server.

- `mdasCollectionName`: default collection name
- `mdasDomainName`: default domain name
- `srbUser`: default SRB user id
- `srbHost`: default host IP of SRB server
- `srbPort`: default host port of SRB server
- `defaultResource`: default storage resource name
- `AUTH_SCHEME`: default authentication mechanism: `PASSWD_AUTH`, `GSI_AUTH`, and `ENCRYPT1`

- `SERVER_DN`: server DN of proxy to invoke SRB server in case of GSI authentication

SRB Authentication

SRB server accepts ENCRYPT1 or GSI authentication. If `password` exists, it authenticates to SRB server via ENCRYPT1 mechanism. Otherwise, by default, it uses GSI authentication. Because the server DN of proxy to invoke SRB server must be specified in case of GSI authentication, the server DN could be read via `-s`, `-ss`, and `-ds` among options of `globus-url-copy`. If the option is not specified, the server DN should be described by `SERVER_DN` in `~/.srb/.MdasEnv` file.

1.2.7 Client Tools

GRASP provides client tools. They provide three user interfaces: a command line interface (CLI), a graphic user interface (GUI), and web interface (WI). They have same functionality for creating and modifying JRDL for a job, submitting the job to JSS, and controlling and monitoring the job. The detail usage of these client tools will appear in another document about GRASP, users' guide.

Command Line Interface (CLI): `grasprun`

The CLI lets you execute commands via `grasprun` at the shell prompt. The CLI could be run at Linux or Windows environment. The user should install the library and certificates to use `grasprun` and know the syntax of JRDL to describe a job.

Graphic User Interface (GUI)

The GUI is a graphic interface based on Java SWING, which is OS independent. While the user must write a JRDL manually in case of `grasprun`, GUI provides convenient interface to load and write the JRDL.

Web Interface (WI): Job Submission Portlet

The WI is a web interface based on Gridsphere, which provides an open-source portlet based Web portal. While the user should install the library and certificates in case of both CLI and GUI, he/she does not have to care about installation as well as the syntax of JRDL in web interface. We have implemented Job submission portlet enabling the user to easily make a JRDL, load the JRDL, submit a job to JSS, and monitor the submitted job.

1.3 Getting help

If you have questions about GRASP or have found a bug, please send an email to kmi@moredream.org. For up-to-date information about GRASP, please visit the web site <http://kmi.gridcenter.or.kr/>.

2 Installation and Configuration

2.1 Support software

2.1.1 Required

- OS: Linux (RedHat 7.3 or more are recommended)
- J2SDK 1.4 (developed under 1.4.2_04, 2.4.2_06)
- ANT (developed under 1.6.2)
- Globus Toolkit 3 (developed under 3.2.1)

Following softwares are required only for RMS and RRS

- MySQL Database (tested under 3.23.49, 4.0.16, and 4.0.20)
- JDBC Driver: MySQL-Connector/J (tested under 3.0.14)
- PHP4 (4.3.4 or latest)

2.1.2 Optional

- Apache Web server (optional, for reservation status monitoring)

2.2 Installing support softwares

2.2.1 Installing Java SDK

Required for: GT3 Webservices components

Recommended Versions: 1.4.x

Download Link: <http://java.sun.com/j2se>

2.2.2 Installing Globus Toolkit

1. Download all source code from <http://www.globus.org>
2. As globus, untar the source installer.
3. Make sure that ANT_HOME and JAVA_HOME are set, and that ant and java are on your PATH.
4. Run

```
# ./install-gt3 /path/to/install
```
7. Configure the Globus Toolkit 3.2, looking through http://www-unix.globus.org/toolkit/docs/3.2/installation/install_config.html

2.2.3 MySQL Database

Download a latest MySQL distribution from <http://www.mysql.com/>.

You can find a copy of MySQL distribution in software archive directory of KISTI Grid Testbed web site : <http://testbed.gridcenter.or.kr/software/index.php?dir=./DBMS/mysql>

Move to a temporary directory and extract the distribution file.

```
# cd /usr/local/src      (download the distribution file in this directory)
# tar zxvf mysql-4.0.16.tar.gz
# cd mysql-4.0.16
```

Configure and compile the source

```
# ./configure --prefix=/usr/local/mysql --with-mysqld-user=root
# make
```

Copy the compiled binaries to the install location.

```
# make install
```

Make a symbolic link for 'mysql' command line client, or add it to the \$PATH variable.

```
# ln -s /usr/local/mysql/bin/mysql /usr/local/bin/mysql
```

Database initialization

```
# /usr/local/mysql/bin/mysql_install_db
```

Start MySQL server daemon

```
# /usr/local/mysql/bin/mysqld_safe -u root &
```

To start MySQL server daemon during system startup, add a line to the rc.local file.

```
# vi /etc/rc.d/rc.local
...
/usr/local/mysql/bin/mysqld_safe -u root &
...
```

Refer to other books or documents about managing and using MySQL.

2.2.4 MySQL-Connector/J 3.0 (JDBC Driver)

Download it from <http://dev.mysql.com/downloads/connector/j/3.0.html>

```
# tar zxvf mysql-connector-java-3.0.14-production.tar.gz
# cp mysql-connector-java-3.0.14-production/mysql-connector-java-
3.0.14-production-bin.jar $JAVA_HOME/jre/lib/ext/
```

2.2.5 PHP4

PHP4 (command line interface) is required for Job Manager Script module in GRASP.

PHP4 compiled with Apache web server is optionally required for monitoring reservation database table.

The libxml2 module must be compiled with PHP.

Download the latest version of libxml2 from <http://xmlsoft.org/sources/>.

```
# tar zxvf libxml2-2.6.16.tar.gz
# cd libxml2-2.6.16/
# ./configure --prefix=/usr/local/libxml2 &> configure.log
# make &> make.log
# make install &> install.log
```

The zlib library must be installed.

Download a latest PHP distribution from <http://www.php.net/>

```
# cd /usr/local/src
# tar zxvf php-4.3.9.tar.gz
# cd php-4.3.9
# ./configure \
--enable-pcntl \
--with-dom=/usr/local/libxml2 --with-zlib-dir=/usr --disable-cgi
# make clean
# make
# make install
```

The configure option '--enable-pcntl', which is for process control in PHP, is required in job manager script module. The '--with-dom' is required for XML processing in PHP.

The '--with-zlib-dir' is required for libxml2 in PHP.

2.2.6 PHP4 + Apache Web Server (optional)

Apache should be configured before compiling PHP.

Download a latest apache distribution from <http://www.apache.org/>

```
# cd /usr/local/src
# tar zxvf apache_1.3.33.tar.gz
# cd apache-1.3.33/
# ./configure
```

Download a latest PHP distribution from <http://www.php.net/>

```
# cd /usr/local/src
# tar zxvf php-4.3.9.tar.gz
# cd php-4.3.9
# ./configure --with-apache=../apache_1.3.33/ \
  --with-config-file-path=/etc/httpd \
  --with-mysql=/usr/local/mysql --enable-pcntl \
  --with-dom=/usr/local/libxml2 --with-zlib-dir=/usr --disable-cgi
# make clean
# make
# make install
```

Compile the Apache web server and install it.

```
# cd /usr/local/src/apache-1.3.33/
# ./configure --prefix=/usr/local/apache \
  --activate-module=src/modules/php4/libphp4.a
# make clean
# make
# make install
```

Setup the PHP installation

```
# cd /usr/local/src/php-4.3.9
# mkdir /etc/httpd; cp php.ini-dist /etc/httpd/php.ini
```

Setup the Apache web server

```
# vi /usr/local/apache/conf/httpd.conf
...
LINE 808(approx.): add a line
# PHP
AddType application/x-httpd-php .php
</IfModule>
...
:wq
```

{Start | stop | restart} the apache web server

```
# /usr/local/apache/bin/apachectl {start | stop | restart}
```

2.2.7 OpenPBS and Cluster Configuration

Computing nodes in a cluster should be configured for rsh and ssh. The job manager script module uses ssh for executing remote program in other computing nodes in the cluster. The rsh have a problem to be used for this purpose. To configure ssh add host keys of all the computing nodes to /etc/ssh/ssh_known_hosts of each computing nodes. The hostname in the known_hosts file should be fully qualified domain name(FQDN).

2.3 Installing GRASP

JSS enables a Grid job to submit to resources managed by RMS. This service could be installed at Linux based server. Even though JSS could be deployed with RMS at the same service container, we recommend separate installation.

RMS enables a Grid job to allocate resources and be executed at computational nodes in local resource. This service could be deployed at front node of each Linux based cluster.

In addition, we provide two kinds of packages: one is a package containing both a command line interface (CLI) and a graphic user interface (GUI), and the other is a package providing a web interface (WI) based on Gridsphere portlet.

Download

<http://kmi.moredream.org/downloads/index.php>

You can download the whole package of GRASP from the web site written above. And then you can get following files of each components of GRASP.

```
Grasp-0.9
|-- jobsubmissioin-0.3.tar.gz
|-- mrmfs-0.3.tar.gz
|-- gridscheduling-0.3-src.tar.gz
|-- globus_gass_copy-srb-0.1.tar.gz
|-- grasp-client-0.2.tar.gz
`-- jobsubmission-portlet-0.3.tar.gz
```

2.3.1 JSS (Job Submission Service)

Installation

As the globus container administrator's account,

```
$ tar zxvf jobsubmission-0.3.tar.gz
$ cd ./jobsubmission
$ ./install-gt3-jobsubmission $GLOBUS_LOCATION
$ su -
# $GLOBUS_LOCATION/bin/setperm.sh
```

Configuration

1. Check if there are hosts to be installed resource manager system in /etc/hosts file.

2.3.2 GSS (Grid Scheduling Service)

Installation

As the globus container administrator's account,

```
$ tar zxvf gridscheduling-0.1-src.tar.gz
$ cd ./gridscheduling-0.1
$ ./gss-install $GLOBUS_LOCATION
```

Configuration**\$GLOBUS_LOCATION/etc/base-info-service.xml**

This file contains the information of a Grid information service which GSS will contact to query resource information. The administrator has to indicate the address of the information service, service data name of resource information, and namespace of the service data.

2.3.3 RMS (Resource Manager Service) and RRS (Resource Reservation Service)

Because both RMS and RRS must be installed simultaneously on same container, both services are packaged to one file: mrmfs-0.11.tar.gz

Required software

- MySQL Database (tested under 3.23.49, 4.0.16, and 4.0.20)
- PHP4 (4.3.4 or latest)
- JDBC Driver: MySQL-Connector/J (tested under 3.0.14)

Optional software

- PHP4 + Apache Web server (for reservation status monitoring)
- OpenPBS (Portable Batch System)
- SRB enabled globus-url-copy

Installation of SRB enabled globus-url-copy

```
$ tar zxvf globus_gass_copy-srb-0.1.tar
$ cd globus_gass_copy-srb
$ mkdir /usr/local/srb
$ tar zxvf SRB3_2_1e.tar.gz -C /usr/local/srb
$ export SRB_LOCATION=/usr/local/srb/SRB3_2_1e
$ ./install.sh
```

Configuration of SRB enabled globus-url-copy

You should edit the `~/.srb/.MdasEnv` file to use SRB server. The `~/.srb/.MdasEnv` file includes following default configuration information to connect to SRB server.

- `mdasCollectionName`: default collection name
- `mdasDomainName`: default domain name
- `srbUser`: default SRB user id
- `srbHost`: default host IP of SRB server
- `srbPort`: default host port of SRB server
- `defaultResource`: default storage resource name
- `AUTH_SCHEME`: default authentication mechanism: `PASSWD_AUTH`, `GSI_AUTH`, and `ENCRYPT1` (You have to specify `AUTH_SCHEME`.)
- `SERVER_DN`: server DN of proxy to invoke SRB server (If you choose `GSI_AUTH` for `AUTH_SCHEME`, you should specify.)

Testing of SRB enabled globus-url-copy

After configuring the above instructions, you should be able to execute `globus-url-copy`.

As the globus container administrator's account,

```
$ grid-proxy-init
$ globus-url-copy \
srb://username.userdomain@IPADDRESS/collectaioname/filename \
file:///tmp/filename
$ more /tmp/filename
```

Installation of RMS and RRS

As the globus container administrator's account,

```
$ tar zxvf mrmfs-0.3.tar.gz
$ cd ./mrmfs
$ ./install-gt3-mrmfs $GLOBUS_LOCATION
$ su -
# $GLOBUS_LOCATION/bin/setperm.sh
```

Configuration of RMS and RRS

After building the service, we should do some configurations.

(1) Configuring Database

We provide SQL files to create or drop database in `$GLOBUS_LOCATION/etc/reservation-sql`.

Create a database named 'moredream' and make tables:

```

$ mysql [-u user] [-p]
mysql> create database moredream;
mysql> quit
$ cd $GLOBUS_LOCATION/etc/reservation-sql
$ mysql [-u user] [-p] moredream < create_tables.sql

```

Refer to other documents to use mysql command.

(2) Edit configuration file

Open `$GLOBUS_LOCATION/etc/reservation.conf` and edit the database connection values:

```

$ vi $GLOBUS_LOCATION/etc/reservation.conf
#
# reservation.conf
###
### database connection
###
dbhost=hostname.example.com
dbport=3306
dbuser=root
dbpass=password
dbname=moredream
###
### reservation service
###
res.resid_prefix=hostname.example.com
res.interval=3000
res.total_nodes=10
# 86400 = 3600*24    = 1 day
# 604800 = 3600*24*7 = 1 week
# 10800 = 3600 * 3   = 3 dyas
res.default_duration=1800
res.default_start_before=10800
res.max_duration=86400
res.start_not_before=60
res.start_not_after=604800

```

Starting and Testing of RMS and RRS

Now, you are ready to start reservation service. Start the globus container.

```

$ globus-start-container
...
http://...:8080/ogsa/services/base/grasp/ReservationService
...

```

To test if RRS is correctly deployed and configured, run a test client program:

```

$ $GLOBUS_LOCATION/bin/test-rrs
http://127.0.0.1:8080/ogsa/services/base/grasp/ReservationService test
Database connection was successful

```

The service is correctly deployed

2.3.4 Client Tools

A client package containing both the CLI and the GUI: **grasp-client-0.1.tar.gz**

This package could be run on both Windows and Linux. This package is also included in JSS package.

Required

- Globus Toolkit 3.2.1 WS Core (<http://www-unix.globus.org/toolkit/downloads/3.2.1/#core>)

Download

<http://kmi.moredream.org/downloads/index.php>

Linux Installation

Note: Before installing, you must set environment variable GLOBUS_LOCATION and copy grasp-client-0.1.tar.gz to \$GLOBUS_LOCATION.

```
$ cd $GLOBUS_LOCATION
$ tar zxvf grasp-client-0.2.tar.gz
```

Linux Configuration

You can specify default factory address in configuration file: \$HOME/.globus/.grasprun.
factory=http://ipaddress:port

Windows Installation

Note: Before installing, you should set environment GLOBUS_LOCATION

1. unzip grasp-client-0.2.zip to %GLOBUS_LOCATION%

Windows Configuration

You can specify default factory address in configuration file: %HOME%\globus\grasprun, where %HOME% is home directory. In case of Windows XP and 2000, home directory is "C:\Documents and Settings\username\
factory=http://ipaddress:port

A Job submission portlet package providing the WI: **jobsubmission-**

portlet-0.1.tar.gz**Required**

- Gridsphere 2.0.1
- Gridportlets portlet

Download

<http://kmi.moredream.org/downloads/index.php>

Installation

```
$ cd $GRIDSPHERE_LOCATION/projects
$ tar zxvf jobsubmission-portlet-0.3.tar.gz
$ cd jobsubmission-portlet
$ ant install
```

2.4 Testing

2.4.1 Running the first job

Now you can test that the services works properly with a simple job. This example executes a single job to echo some arguments by “FORK” local job manager. It determines the resources not by GSS but by user’s assignment. Therefore, you need to edit the string `rms_machine` in `$GLOBUS_LOCATION/schema/base/grasp/jobsubmission/examples/fork_single.xml` to be actual hostname deployed RMS to wish to run a job. You should check if there is hostname of installed job submission machine in `/etc/hosts` file. Here we use a job submission client “`grasprun`” to make a job submitted to JSS.

```
$ grid-proxy-init
$ grasprun -factory
http://jobsubmission_machine:8080/ogsa/services/base/grasp/JobSubmissionFactoryService -file
$GLOBUS_LOCATION/schema/base/grasp/jobsubmission/examples/fork_single.xml
```

Note: If you have configured the factory address in `~/.globus/.grasprun` file, you do not have to specify `-factory` option.

If you want to make RMS use “PBS” instead of “FORK”, you should try `pbs_single.xml` instead of `fork_single.xml`

```
$ grasprun -factory
http://jobsubmission_machine:8080/ogsa/services/base/grasp/JobSubmissionFactoryService -file
$GLOBUS_LOCATION/schema/base/grasp/jobsubmission/examples/pbs_single.xml
```

If you want to automatically determine resources by GSS, you should try `single.xml`.

```
$ grasprun -factory
http://jobsubmission_machine:8080/ogsa/services/base/grasp/JobSubmissionFactoryService -file
$GLOBUS_LOCATION/schema/base/grasp/jobsubmission/examples/single.xml
```

3 Operation of GRASP services

As you might get an intuition where to install and use each service in GRASP from the architecture, each service of GRASP has to be properly installed and operated. Please understand the Figure 1 and the role of each service before deployment. Both JSS and GSS do not have to be installed together on a same computing node and they could be operated on the same machine or separated.

Followings are what you have to know to operate services properly.

3.1 JSS

JSS enables a Grid job to submit to resources by using RMS. This service could be installed at Linux based server. Even though JSS could be deployed with RMS at the same service container, we recommend separate installation.

3.1.1 Logging

JSS provide logging support by adding following three lines to **\$GLOBUS_LOCATION/ogsilogging.properties** file where jobsubmission.log is a target file to append the log for job submission.

```
org.moredream.ogsa.impl.base.grasp.jobsubmission.JobSubmissionThread=jobsubmission.log,info
```

```
org.moredream.ogsa.impl.base.grasp.jobsubmission.JobSubmissionImpl=jobsubmission.log,info
```

```
org.moredream.ogsa.impl.base.grasp.jobsubmission.MultipleJobSubmissionThread=jobsubmission.log,info
```

3.2 GSS

\$GLOBUS_LOCATION/etc/sched-plugin-conf.xml

This file provides information of the scheduling plugins which is included in the GSS. GSS can have several scheduling plugins. Whenever a plugin is added, the administrator has to add required information in this file. The contents of this file will be provided to users in the form of service data of GSFS (Grid Scheduling Factory Service).

3.3 RMS

RMS enables a Grid job to allocate resources and be executed at computational nodes in local resource. This service could be deployed at front node of each Linux based cluster. If you want to use SRB server at machine installed RMS, you should install the SRB enabled globus-url-copy.

3.3.1 Logging

RMS provide logging support by adding following four lines to **\$GLOBUS_LOCATION/log4j.properties** file. Logging information will be appended to `~/.globus/uhe-hostname/log` file of each local account.

```
log4j.category.org.moredream.ogsa.impl.base.grasp.rms.jobmanager.ResourceManagerImpl=DEBUG
```

```
log4j.category.org.moredream.ogsa.impl.base.grasp.rms.jobmanager.JobManager=DEBUG
```

```
log4j.category.org.moredream.ogsa.impl.base.grasp.rms.jobmanager.JobManagerScript=DEBUG
```

```
log4j.category.org.moredream.ogsa.impl.base.grasp.rms.jobmanager.JobExecutionTimeHelper=DEBUG
```

3.4 RRS

RRS has one configuration file related to database connection and one module to help monitoring the reservation status in `$GLOBUS_LOCATION/etc` directory.

3.4.1 Logging

RRS provide logging support by adding following two lines to **\$GLOBUS_LOCATION/ogsilogging.properties** file where `reservation.log` is a target file to append the log for resource reservation.

```
org.moredream.ogsa.impl.base.grasp.reservation.impl.ReservationProvider=reservation.log,trace
```

```
org.moredream.ogsa.impl.base.grasp.reservation.impl.ReservationManagerThread=reservation.log,trace
```

3.4.2 Database Connection

Using `$GLOBUS_LOCATION/etc/reservation.conf`, you can configure the database connection for RSS.

```

#
# reservation.conf
###
### database connection
###
dbhost=hostname.example.com
dbport=3306
dbuser=root
dbpass=password
dbname=moredream
###
### reservation service
###
res.resid_prefix=hostname.example.com
res.interval=3000
res.total_nodes=10
# 86400 = 3600*24    = 1 day
# 604800 = 3600*24*7 = 1 week
# 10800 = 3600 * 3   = 3 dyas
res.default_duration=1800
res.default_start_before=10800
res.max_duration=86400
res.start_not_before=60
res.start_not_after=604800

```

3.4.3 Monitoring Reservation Status

Using `$GLOBUS_LOCATION/etc/reservation-dumpdb` module included in the distribution, you can monitor the reservation status.

For that, you need to install Apache + PHP4 to a web server host.

Extract the distribution file under the web root directory (e.g. `/usr/local/apache/htdocs`) in your web server. Edit the `config.php` to configure database connection parameters:

```

$ vi dumpdb/config.php
...

# database connection
$conf['dbhost'] = "localhost";
$conf['dbuser'] = "root";
$conf['dbpasswd'] = "";
$conf['dbname'] = "moredream";
...

```

And open `dumpdb.php` using your web browser.

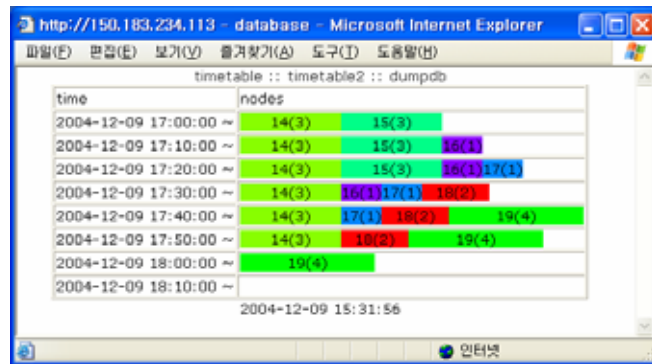


Figure 9. Monitoring reservation status using web browser

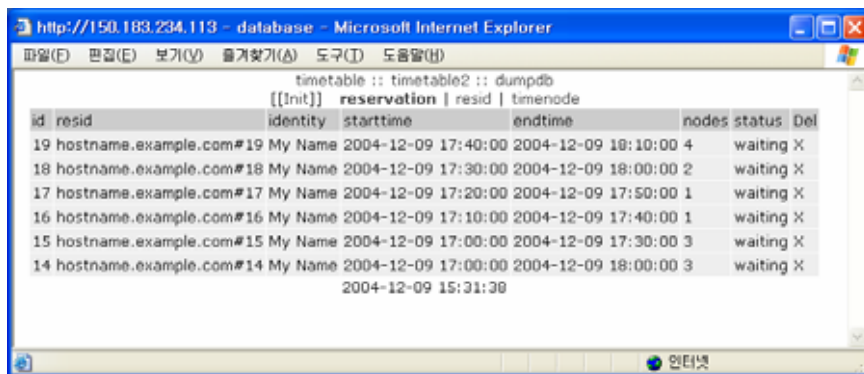


Figure 10. Database table of reservation items

4 References

<http://www-unix.globus.org/toolkit/docs/3.2/installation/index.html>

<http://java.sun.com/j2se/1.4.2/index.jsp>

<http://ant.apache.org/manual/index.html>

http://testbed.gridcenter.or.kr/software/OpenPBS/doc/v2.3_admin.pdf

<http://www.gridisphere.org/gridsphere/docs/index.html>

MySQL (<http://www.mysql.com/>)

PHP (<http://www.php.net/>)